ModelSim®

Xilinx Edition II

Tutorial

Version 5.7c

Published: 11/Mar/03

The world's most popular HDL simulator

ModelSim support

Support for ModelSim is available from your FPGA vendor. See the About ModelSim dialog box (accessed via the Help menu) for contact information.

Table of Contents

Introduction

Chapter contents

Software versions

This documentation was written to support ModelSim 5.7c for Microsoft Windows 98/Me/NT/2000/XP. If the ModelSim software you are using is a later release, check the README file that accompanied the software. Any supplemental information will be there.

Although this document covers both VHDL and Verilog simulation, you will find it a useful reference even if your design work is limited to a single HDL.

Standards supported

ModelSim VHDL supports both the IEEE 1076-1987 and 1076-1993 VHDL, the 1164-1993 *Standard Multivalue Logic System for VHDL Interoperability*, and the 1076.2-1996 *Standard VHDL Mathematical Packages* standards. Any design developed with ModelSim will be compatible with any other VHDL system that is compliant with either IEEE Standard 1076-1987 or 1076-1993.

ModelSim Verilog is based on IEEE Std 1364-1995 and a partial implementation of 1364-2001 (see */<install_dir>/modeltech/docs/technotes/vlog_2001.note* for implementation details) *Standard Hardware Description Language.* The Open Verilog International *Verilog LRM version 2.0* is also applicable to a large extent. Both PLI (Programming Language Interface) and VCD (Value Change Dump) are supported for ModelSim PE and SE users.

In addition, all products support SDF 1.0 through 3.0, VITAL 2.2b, VITAL'95 – IEEE 1076.4-1995, and VITAL 2000 – IEEE 1076.4-2000.

Assumptions

We assume that you are familiar with the use of your operating system. If you are not familiar with Microsoft Windows, we recommend that you work through the tutorials provided with MS Windows before using Model*Sim*.

We also assume that you have a working knowledge of VHDL and Verilog. Although ModelSim is an excellent tool to use while learning HDL concepts and practices, this document is not written to support that goal.

Before you begin

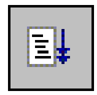
Preparation for some of the lessons leaves certain details up to you. You will decide the best way to create directories, copy files and execute programs within your operating system. (When you are operating the simulator within ModelSim's GUI, the interface is consistent for all platforms.)

Additional details for VHDL and Verilog simulation can be found in the *ModelSim User's Manual* and *Command Reference*.

Command, button, and menu equivalents

Many of the lesson steps are accomplished by a button or menu selection. When appropriate, VSIM command line (PROMPT:) or menu (MENU:) equivalents for these selections are shown in parentheses within the step. This example shows three options to the **run -all** command, a button, prompt command, and a menu selection.

(PROMPT: run -all) (MENU: Simulate > Run > Run -All)



Drag and drop

Drag and drop allows you to copy and move signals among windows. If drag and drop applies to a lesson step, it is noted in a fashion similar to MENUS and PROMPTS with: DRAG&DROP.

Command history

As you work on the lessons, keep an eye on the Main transcript window. The commands invoked by buttons and menu selections are echoed there. You can scroll through the command history with the up and down arrow keys, or the command history may be reviewed with several shortcuts at the ModelSim/VSIM prompt.

Shortcut	Description
click on prompt	left-click once on a previous ModelSim or VSIM prompt in the transcript to copy the command typed at that prompt to the active cursor
`his or history`	shows the last few commands (up to 50 are kept)

Reusing commands from the Main transcript

ModelSim's Main transcript can be saved, and the resulting file used as a DO (macro) file to replay the transcribed commands. You can save the transcript at any time before or during simulation. You have the option of clearing the transcript (File > Transcript > Clear Transcript) if you don't want to save the entire command history.

To save the contents of the transcript select **File > Transcript > Save Transcript As** from the Main menu.

Replay the saved transcript with the **do** command:

```
do <do file name>
```

For example, if you saved a series of compiler commands as *mycompile.do* (the .do extension is optional), you could recompile with one command:

```
do mycompile.do
```

▶ **Note:** Neither the prompt nor the Return that ends a command line are shown in the examples.

Lesson 1 - Creating a Project

The goals for this lesson are:

- Create a project

A project is a collection entity for an HDL design under specification or test. Projects ease interaction with the tool and are useful for organizing files and specifying simulation settings. At a minimum, projects have a work library and a session state that is stored in a .mpf file. A project may also consist of:

- HDL source files or references to source files
- other files such as READMEs or other project documentation
- local libraries
- references to global libraries

For more information about using project files, see the *ModelSim User's Manual*.

Creating a project

1 Start ModelSim with one of the following:

for Windows - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

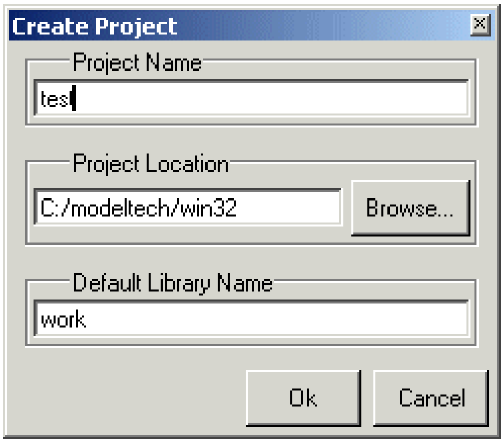
```
modelsim.exe
```

Note: If you didn't add ModelSim to your search path during installation, you will have to include the full path when you type this command at a DOS prompt.

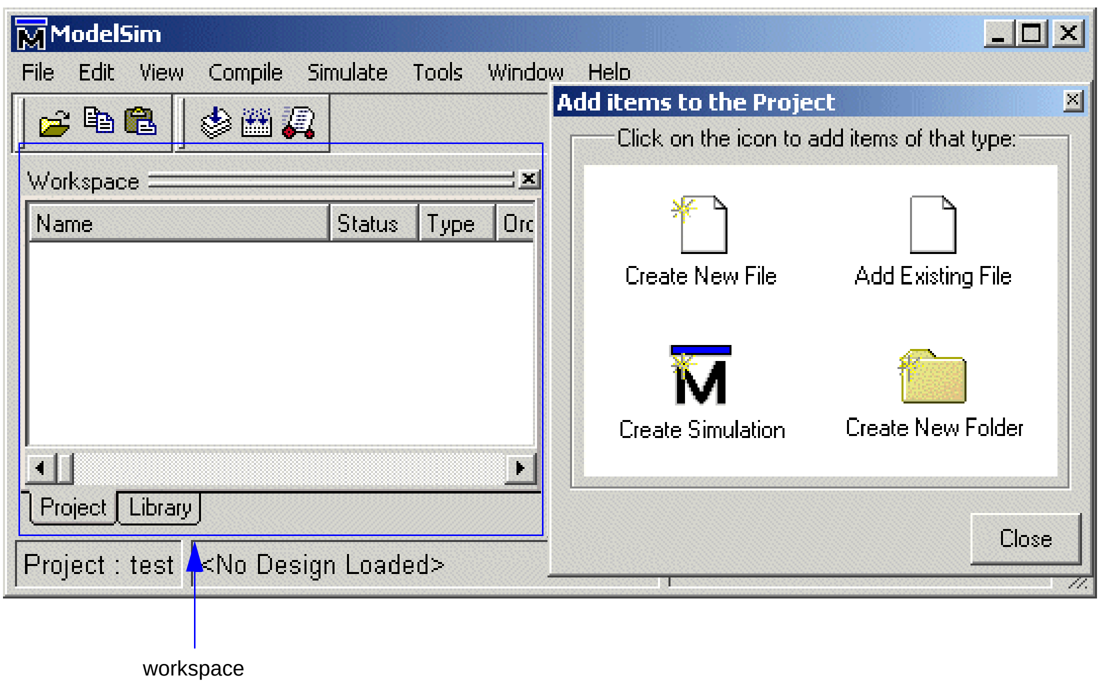
Upon opening ModelSim for the first time, you will see the **Welcome to ModelSim** dialog. (If this screen is not available, you can display it by selecting **Help > Welcome Menu** from the Main window.)

2 Select **Create a Project** from the Welcome dialog, or **File > New > Project** (Main window). In the **Create Project** dialog, enter "test" as the Project Name and select a

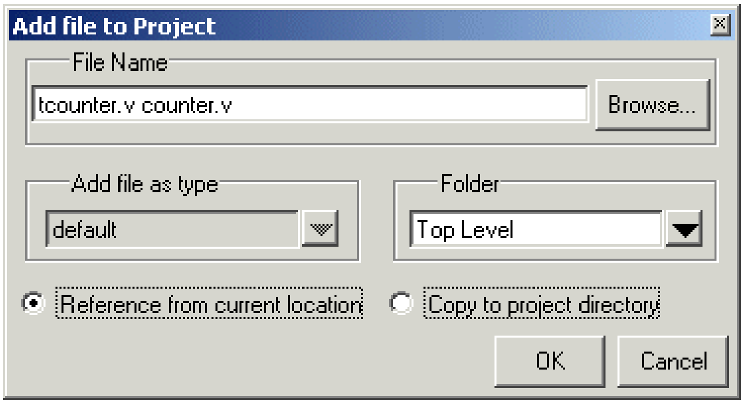
directory where the project file will be stored. Leave the Default Library Name set to "work."



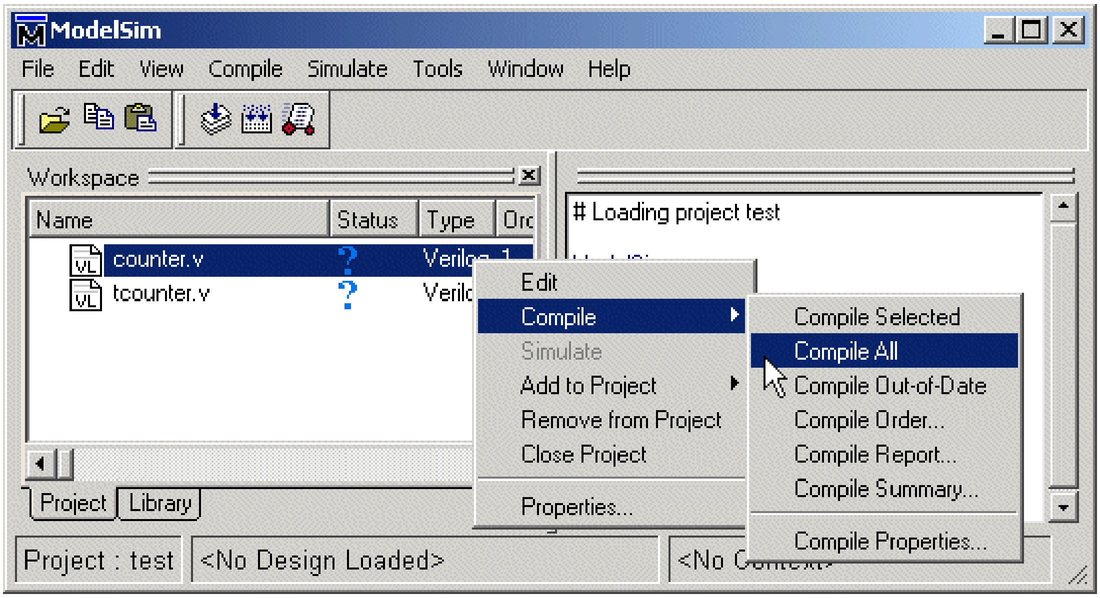
Upon selecting OK, you will see a blank Project tab in the workspace area of the Main window and the **Add Items to the Project** dialog.



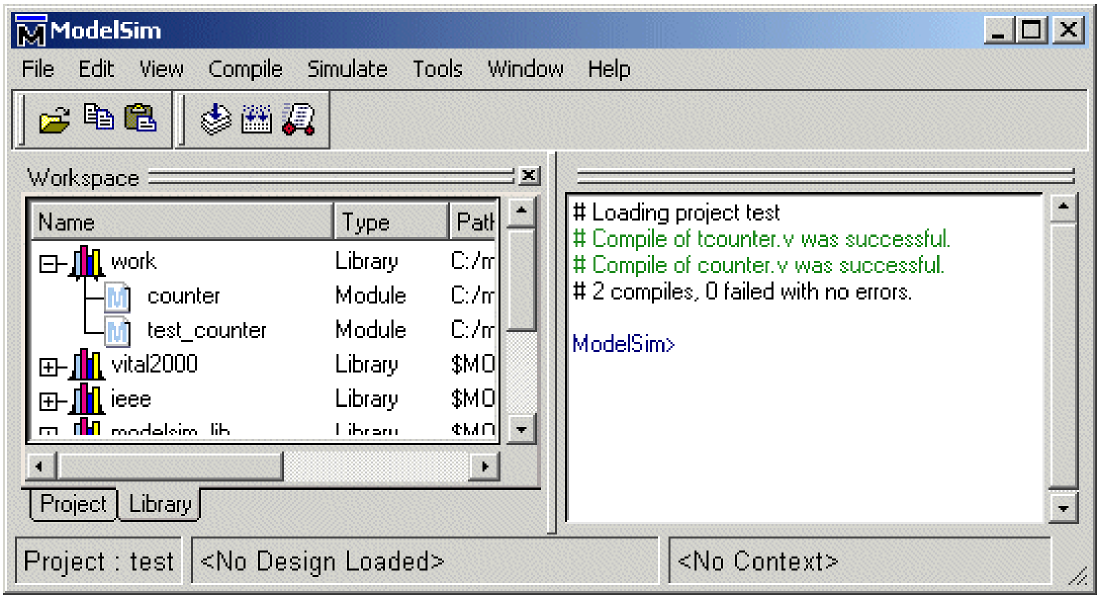
3 The next step is to add the files that contain your design units. Click **Add Existing File** in the **Add items to the Project** dialog. For this exercise, we'll add two Verilog files. Click the **Browse** button in the **Add file to Project** dialog and open the examples directory in your ModelSim installation. Select *tcounter.v* and *counter.v*. Select **Reference from current location** and then click OK. Close the **Add items to the Project** dialog.



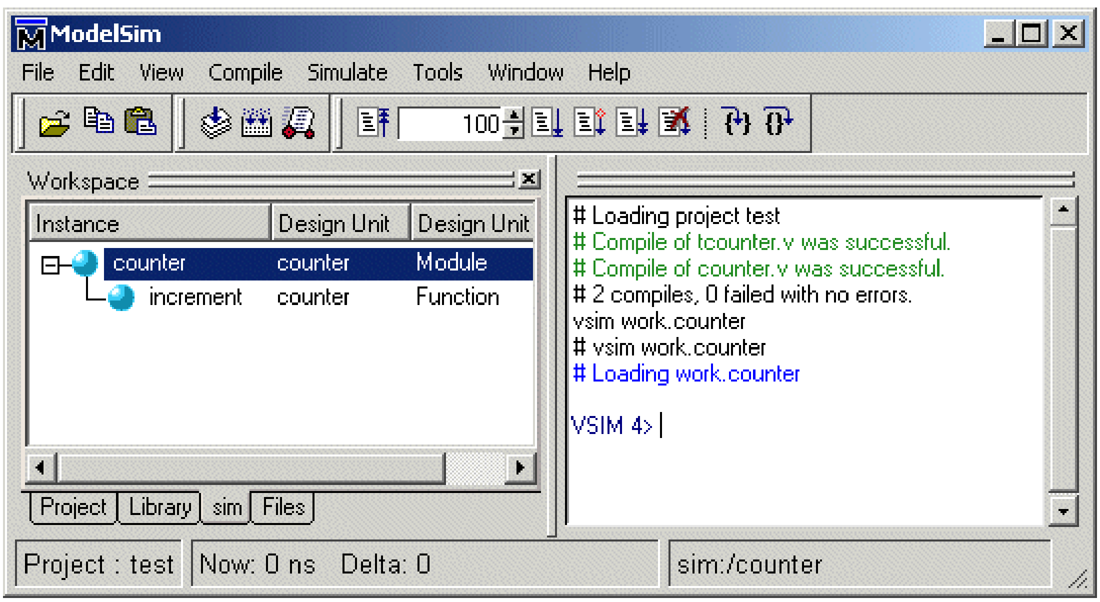
4 Click your right mouse button (2nd button in Windows; 3rd button in UNIX) in the Project page and select **Compile > Compile All**.



5 The two files are compiled. Click on the Library tab and expand the *work* library by clicking the "+" icon. You'll see the compiled design units listed.



6 The last step in this exercise is to load one of the design units. Double-click *counter* on the Library page. You'll see a new page appear in the Workspace that displays the structure of the *counter* design unit.



At this point, you would generally run the simulation and analyze or debug your design. We'll do just that in the upcoming lessons. For now, let's wrap up by ending the simulation and closing the project. Select **Simulate** > **End Simulation** and confirm that you want to

quit simulating. Next, select **File > Close > Project** and confirm that you want to close the project.

Note that a *test.mpf* file has been created in your working directory. This file contains information about the project *test* that you just created. You can open this project in future sessions by selecting **File > Open > Project**.

Lesson 2 - Basic VHDL simulation

The goals for this lesson are:

- Create a library and compile a VHDL file
- Load a design
- Learn about the basic ModelSim windows, mouse, and menu conventions
- Force the value of a signal
- Run ModelSim using the **run** command
- Set a breakpoint
- Single-step through a simulation run

The project feature covered in Lesson 1 executes several actions automatically such as creating and mapping work libraries. In this lesson we will go through the entire process so you get a feel for how ModelSim really works.

Compiling the design

1 Start by creating a new directory for this exercise (in case other users will be working with these lessons). Create the directory, then copy all of the VHDL (.vhd) files from *\<install_dir>\modeltech\examples* to the new directory.

Make sure the new directory is the current directory. Do this by invoking ModelSim from the new directory or by selecting **File > Change Directory** (Main window).

2 Start ModelSim with one of the following:

for Windows - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

```
modelsim.exe
```

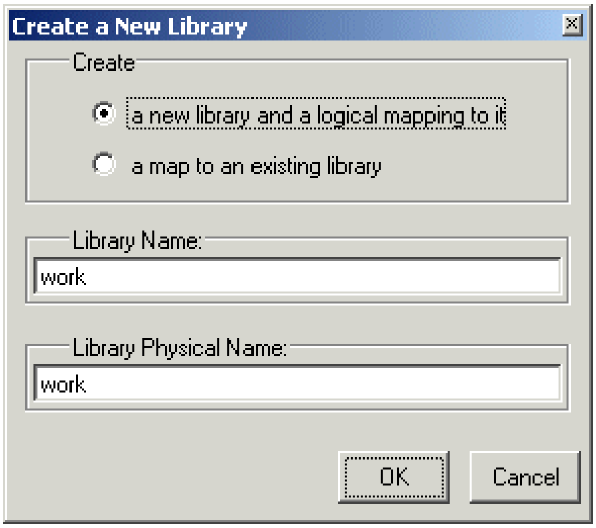
Note: If you didn't add ModelSim to your search path during installation, you will have to include the full path when you type this command at a DOS prompt.

Click **Close** if the Welcome dialog appears.

3 Before you compile any HDL code, you'll need a design library to hold the compilation results. To create a new design library, make this menu selection in the Main window: **File** > **New** > **Library**.

(PROMPT: vlib work
vmap work work)

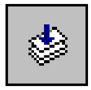
Make sure **Create: a new library and a logical mapping to it** is selected. Type "work" in the Library Name field and then select **OK**.



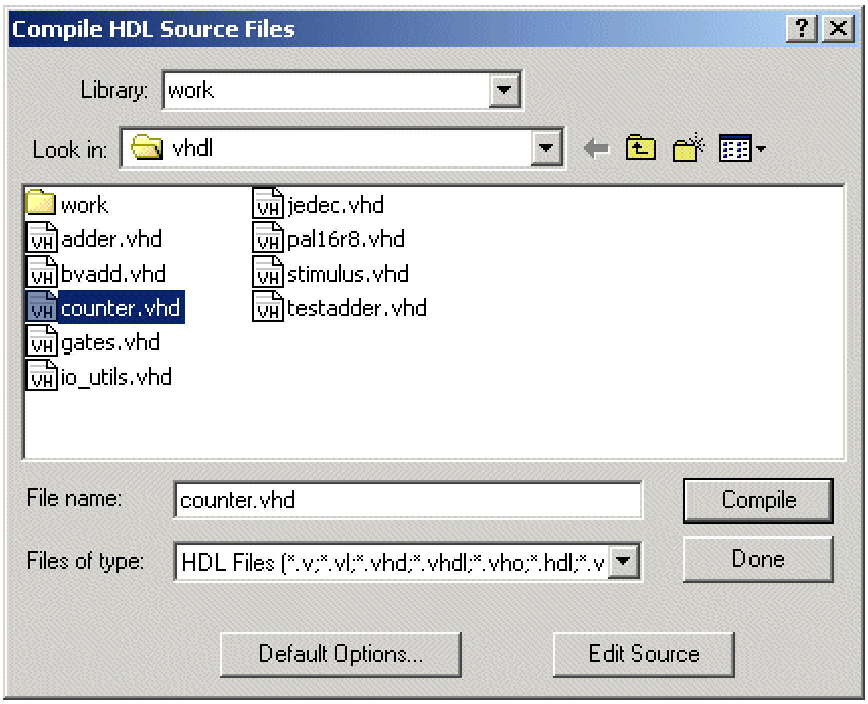
This creates a subdirectory named *work* - your design library - within the current directory. ModelSim saves a special file named *_info* in the subdirectory.

Note: Do not create a Library directory using Windows commands, because the *_info* file will not be created. Always use the File menu or the **vlib** command from either the ModelSim or DOS prompt.

4 Compile the file *counter.vhd* into the new library by selecting **Compile > Compile**.

(PROMPT: vcom counter.vhd) 

This opens the **Compile HDL Source Files** dialog. (You won't see this dialog if you invoke vcom from the command line.)



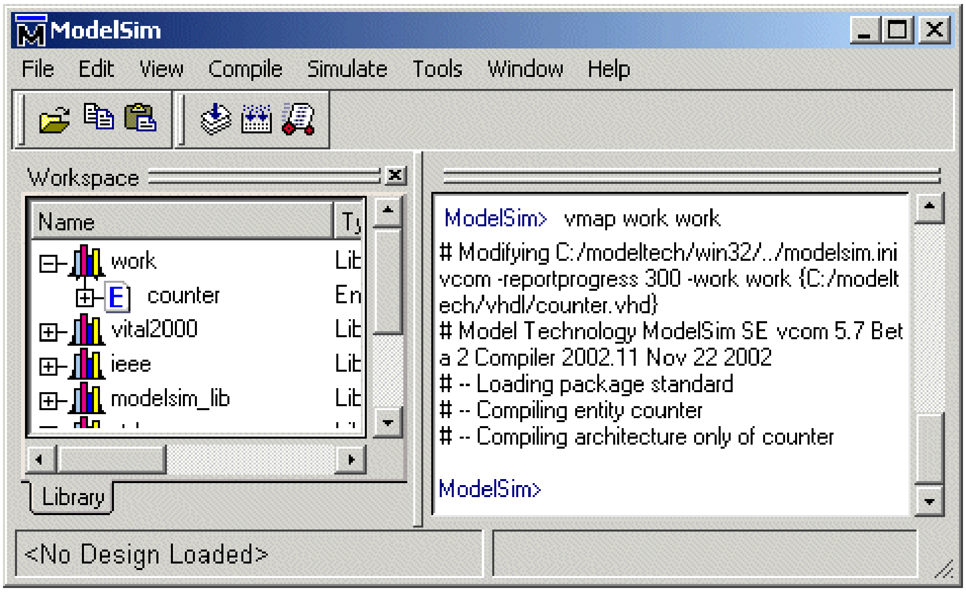
Complete the compilation by selecting *counter.vhd* from the file list and clicking **Compile**. Select **Done** when you are finished.

You can compile multiple files in one session from the file list. Individually select and compile the files in the order required by your design.

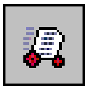
Note that you can have ModelSim determine the compile order. See "Auto-generating compile order" in the Project chapter of the *ModelSim User's Manual* for details.

Loading the design

1 In the Library tab of the Main window Workspace, click the "+" sign next to the 'work' library to see the *counter* design unit.



Double-click *counter* to load the design unit.

(PROMPT: vsim counter) 

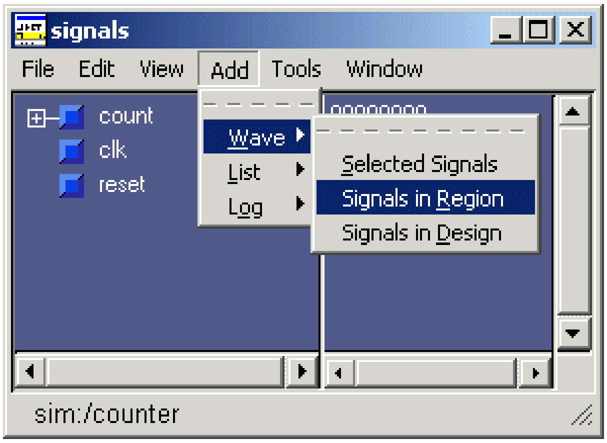
2 Next, select **View** > **All Windows** from the Main window menu to open all ModelSim windows.

(PROMPT: view *)

For descriptions of the windows, consult the *ModelSim User's Manual*.

3 Next let's add top-level signals to the Wave window by selecting **Add** > **Wave** > **Signals in Region** from the Signals window menu.

(PROMPT: add wave /counter/*)



Running the simulation

We will start the simulation by applying stimulus to the clock input.

1 Click in the Main window and enter the following command at the VSIM prompt:

```
force clk 1 50, 0 100 -repeat 100
```

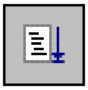
(Signals MENU: Edit > Clock)

ModelSim interprets this **force** command as follows:

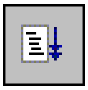
- force clk to the value 1 at 50 ns after the current time
- then to 0 at 100 ns after the current time
- repeat this cycle every 100 ns

2 Now you will exercise two different **Run** functions from the toolbar buttons on either the Main or Wave window. (The **Run** functions are identical in the Main and Wave windows.) Select the **Run** button first. When the run is complete, select **Run -All**.

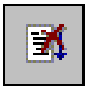
Run. This causes the simulation to run and then stop after 100 ns.

(PROMPT: run 100) (Main MENU: Simulate > Run > Run 100ns) 

Run -All. This causes the simulator to run forever. To stop the run, go on to the next step.

(PROMPT: run -all) (Main MENU: Simulate > Run > Run -All) 

3 Select the **Break** button on either the Main or Wave window toolbar to interrupt the run. The simulator will stop running as soon as it gets to an acceptable stopping point.

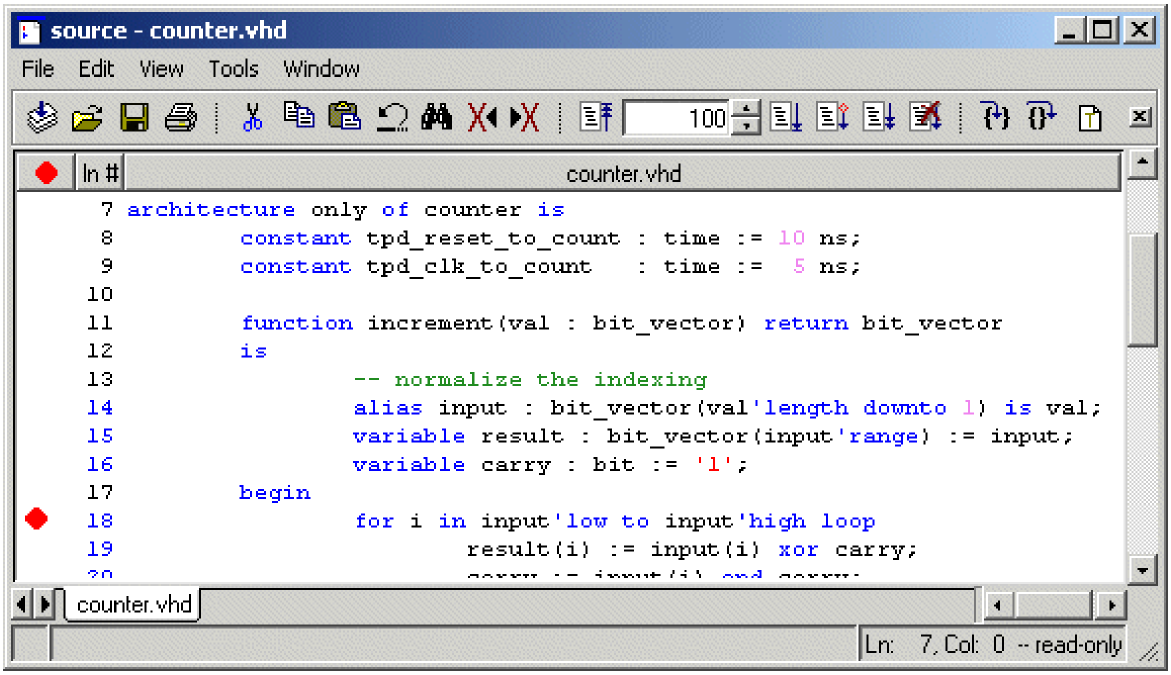
(Main MENU: Simulate > Break) 

The arrow in the Source window points to the next HDL statement to be executed. (If the simulator is not evaluating a process at the time the Break occurs, no arrow will be displayed in the Source window.)

4 Next, you will set a breakpoint in the function on line 18. Scroll the Source window until line 18 is visible. Click on or near line number 18 to set the breakpoint.

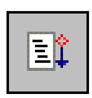
You should see a red dot next to the line number where the breakpoint is set. The breakpoint can be toggled between enabled and disabled by clicking it. When a breakpoint is disabled, the dot appears open. To delete the breakpoint, click the line number with your right mouse button and select Remove Breakpoint 18.

(PROMPT: bp counter.vhd 18)

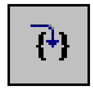


Note: Breakpoints can be set only on lines with blue line numbers.

5 Select the **Continue Run** button to resume the run that you interrupted. ModelSim will hit the breakpoint, as shown by an arrow in the Source window and by a Break message in the Main window.

(PROMPT: run -continue) (MENU: Simulate > Run > Continue) 

6 Click the **Step** button in the Main or Source window several times to single-step through the simulation. Notice that the values change in the Variables window (you may need to expand the Variables window).

(PROMPT: step) (MENU: Simulate > Run > Step) 

7 This concludes the basic VHDL simulation tutorial. When you’re done, quit the simulator by entering the command:

```
quit -force
```

This command exits ModelSim without asking for confirmation.

Lesson 3 - Basic Verilog simulation

The goals for this lesson are:

- Compile a Verilog design
- View signals in the design
- Examine the hierarchy of the design
- Simulate the design
- Change the default run length
- Set a breakpoint

The project feature covered in Lesson 1 executes several actions automatically such as creating and mapping work libraries. In this lesson we will go through the entire process so you get a feel for how ModelSim really works.

Compiling the design

1 Start by creating a new directory for this exercise (in case other users will be working with these lessons). Create the directory, then copy all of the Verilog (.v) files from \<*install_dir*>*\modeltech\examples* to the new directory.

Make sure the new directory is the current directory. Do this by invoking ModelSim from the new directory or by selecting **File > Change Directory** (Main window).

2 Start ModelSim with one of the following:

for Windows - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

```
modelsim.exe
```

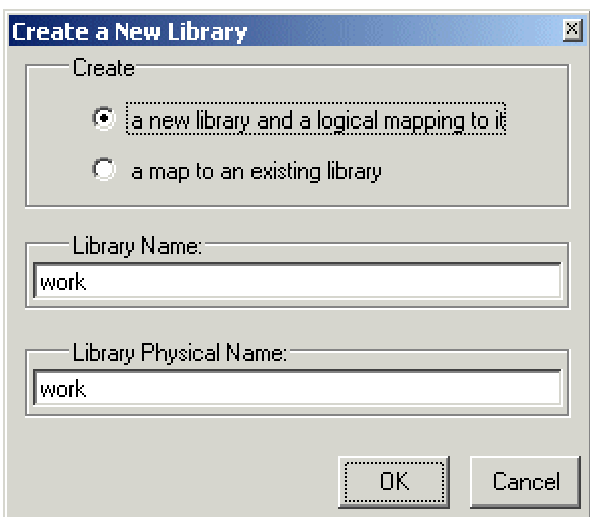
Note: If you didn't add ModelSim to your search path during installation, you will have to include the full path when you type this command at a DOS prompt.

Click **Close** if the Welcome dialog appears.

3 Before you compile any HDL code, you'll need a design library to hold the compilation results. To create a new design library, make this menu selection in the Main window: **File** > **New** > **Library**.

(PROMPT: vlib work
vmap work work)

Make sure **Create: a new library and a logical mapping to it** is selected. Type "work" in the Library Name field and then select **OK**.



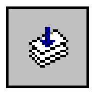
This creates a subdirectory named *work* - your design library - within the current directory. ModelSim saves a special file named *_info* in the subdirectory.

Note: Do not create a Library directory using Windows commands, because the *_info* file will not be created. Always use the File menu or the **vlib** command from either the ModelSim or DOS prompt.

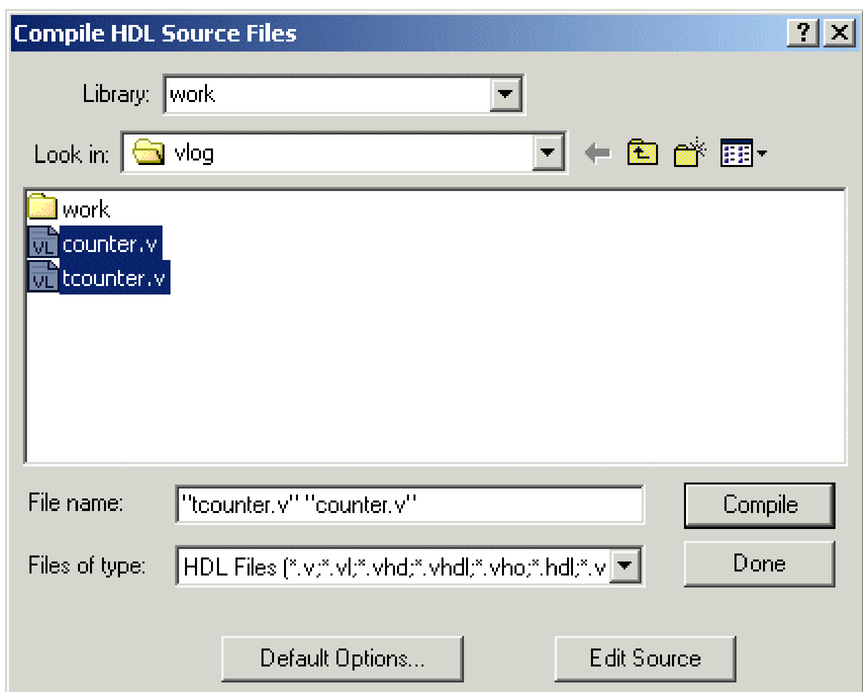
In the next step you'll compile the Verilog design. The example design consists of two Verilog source files, each containing a unique module. The file *counter.v* contains a module called **counter**, which implements a simple 8-bit binary up-counter. The other file, *tcounter.v*, is a testbench module (**test_counter**) used to verify **counter**.

Under simulation you will see that these two files are configured hierarchically with a single instance (instance name **dut**) of module **counter** instantiated by the testbench. You'll get a chance to look at the structure of this code later. For now, you need to compile both files into the **work** design library.

4 Compile the *counter.v*, and *tcounter.v* files into the **work** library by selecting **Compile > Compile** from the Main window menu.

(PROMPT: vlog counter.v tcounter.v) 

This opens the **Compile HDL Source Files** dialog.

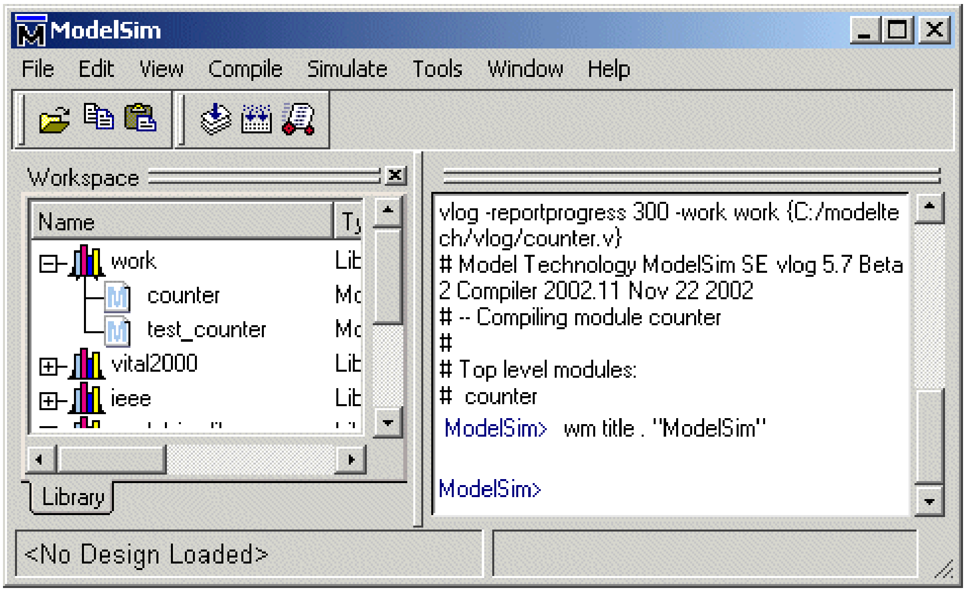


Select *counter.v* and *tcounter.v* (use Ctrl + click) and then choose **Compile** and then **Done**.

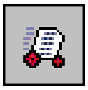
Note: The order in which you compile the two Verilog modules is not important (other than the source-code dependencies created by compiler directives). This may again seem strange to Verilog-XL users who understand the possible problems of interface checking between design units, or compiler directive inheritance. ModelSim defers such checks until the design is loaded. So it doesn't matter here if you choose to compile *counter.v* before or after *tcounter.v*.

Loading the design

1 In the Library tab of the Main window Workspace, click the "+" sign next to the ’work’ library to see the *counter* and *test_counter* design units.



Double-click *test_counter* to load the design unit.

(PROMPT: vsim test_counter) 

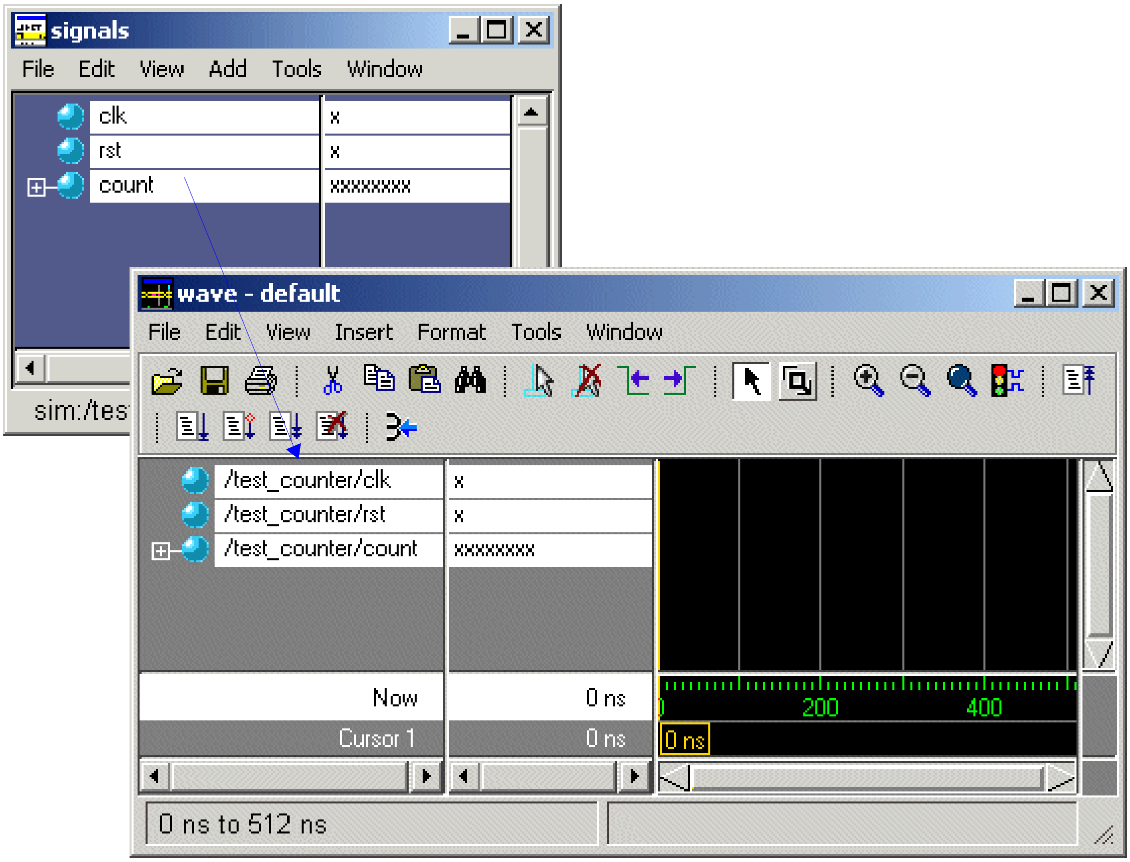
2 Bring up the Signals, Source, and Wave windows by entering the following command at the VSIM prompt within the Main window:

```
view signals source wave
```

(Main MENU: View > <window name>)

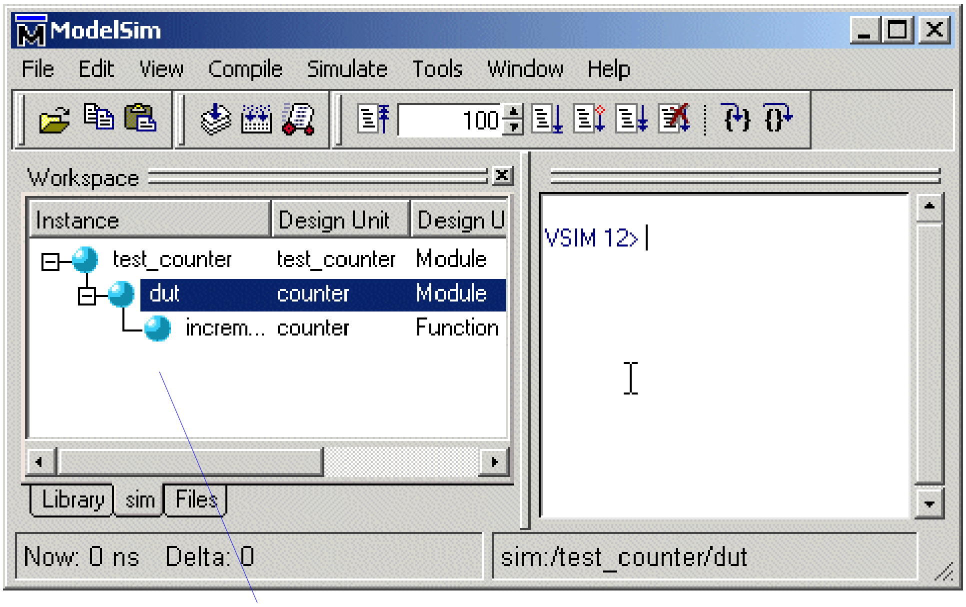
3 Now let's add signals to the Wave window with ModelSim's drag and drop feature.

In the Signals window, select **Edit > Select All** to select the three signals. Drag the signals to either the pathname or the values pane of the Wave window.



HDL items can also be copied from one window to another (or within the Wave and List windows) with the **Edit > Copy** and **Edit > Paste** menu selections.

4 You may have noticed when you loaded the design in Step 1 that a new tab appeared in the Workspace area of the Main window.



Structure pane

The Structure tab shows the hierarchical structure of the design. By default, only the top level of the hierarchy is expanded. You can navigate within the hierarchy by clicking on any line with a "+" (expand) or "-" (contract) symbol. The same navigation technique works anywhere you find these symbols within ModelSim.

By clicking the "+" next to *dut* you can see all three hierarchical levels: *test_counter*, *dut* (counter), and a function called *increment*. (If *test_counter* is not displayed, you simulated *counter* instead of *test_counter*.)

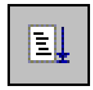
5 Click on *increment* and notice how other ModelSim windows are automatically updated as appropriate. Specifically, the Source window displays the Verilog code at the hierarchical level you selected in the Structure tab, and the Signals window displays the appropriate signals. Using the Structure tab in this way is analogous to scoping commands in interpreted Verilog simulators.

For now, make sure the *test_counter* module is showing in the Source window by clicking on the top line in the Structure tab.

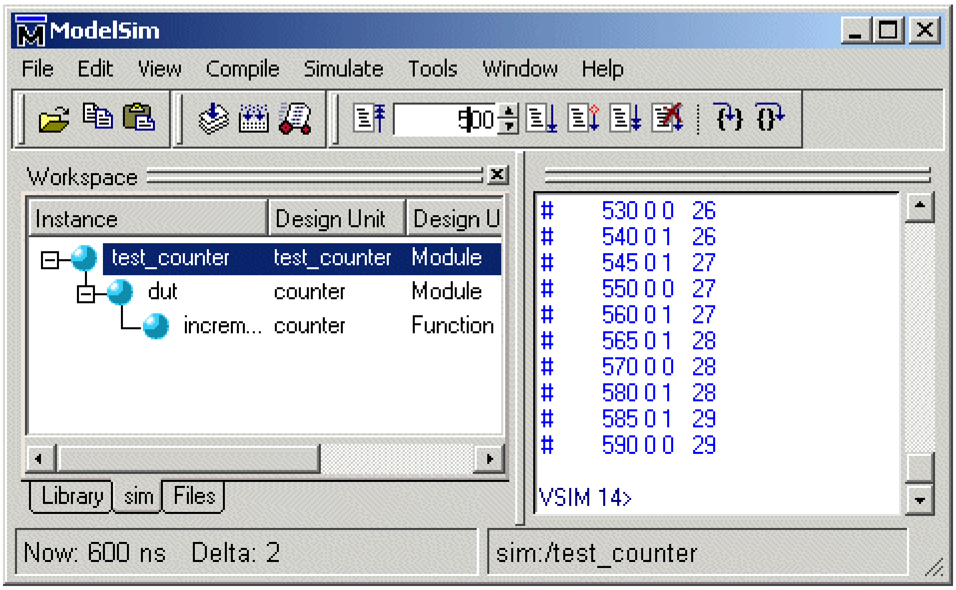
Running the simulation

Now you will exercise different Run functions from the toolbar.

1 Select the **Run** button on the Main window toolbar. This causes the simulation to run and then stop after 100 ns (the default simulation length).

(PROMPT: run) (MENU: Simulate > Run > Run 100 ns) 

2 Next change the run length to 500 on the **Run Length** selector and select the **Run** button again.



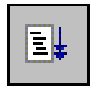
Now the simulation has run for a total of 600ns (the default 100ns plus the 500 you just asked for). The status bar at the bottom of the Main window displays this information.

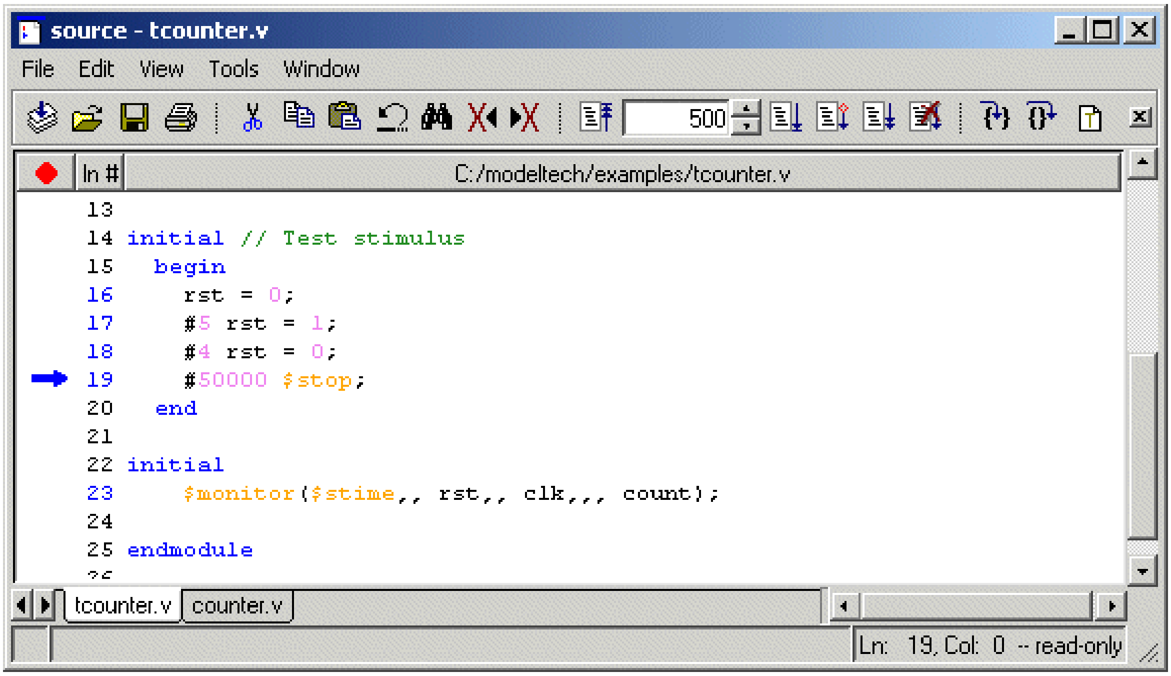
3 The last command you executed (**run 500**) caused the simulation to advance for 500ns. You can also advance simulation to a specific time. Type:

```
run @ 3000
```

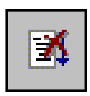
This advances the simulation to time 3000ns. Note that the simulation actually ran for an additional 2400ns (3000 - 600).

4 Now select the **Run -All** button from the Main window toolbar. This causes the simulator to run until the stop statement in *tcounter.v*.

(PROMPT: run -all) (MENU: Simulate > Run > Run -All) 



You can also use the **Break** button to interrupt a run.

(MENU: Simulate > Break) 

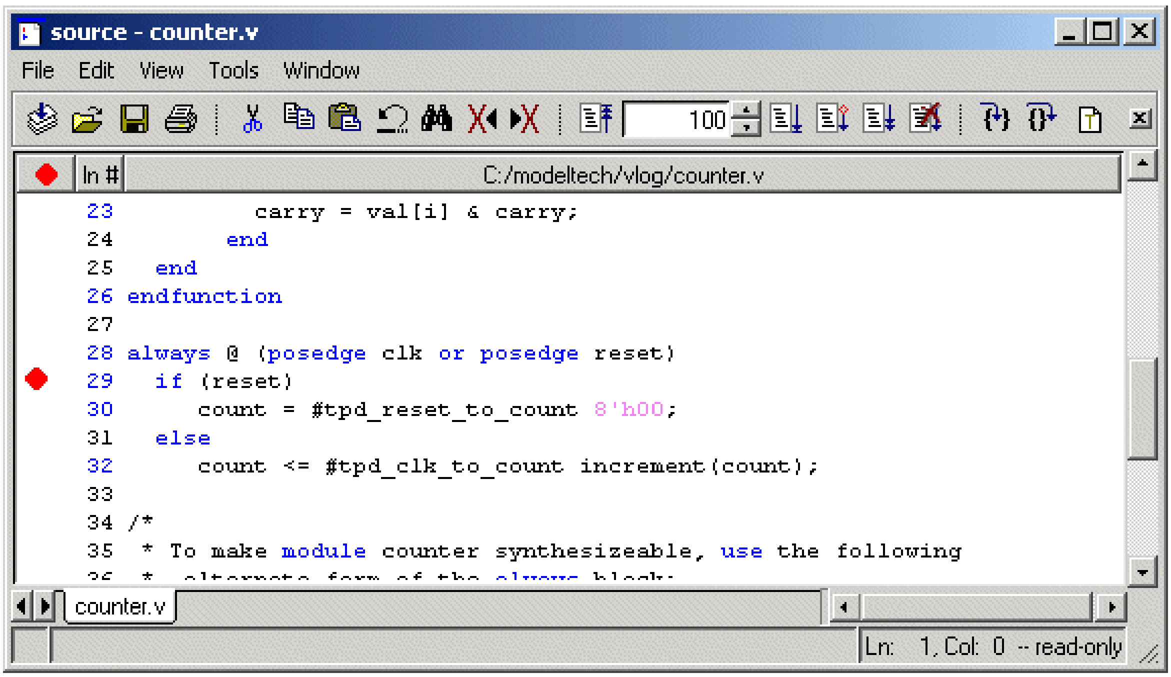
Debugging

Next we'll take a brief look at an interactive debugging feature of the ModelSim environment.

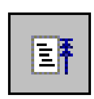
1 Let's set a breakpoint at line 29 in the *counter.v* file (which contains a call to the Verilog function increment). To do this, select *dut* in the Structure tab of the Workspace. Move the cursor to the Source window and scroll the window to display line 29. Click on or to the left of the 29 to set a breakpoint. You should see a red dot next to the line number where the breakpoint is set.

The breakpoint can be toggled between enabled and disabled by clicking it. When a breakpoint is disabled, the dot appears open. To delete the breakpoint, click the line number with your right mouse button and select **Remove Breakpoint 29**.

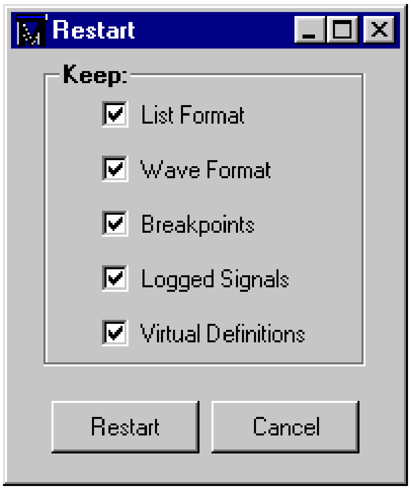
Note: Breakpoints can be set only on lines with blue line numbers.



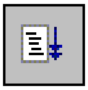
2 Select the **Restart** button to reload the design elements and reset the simulation time to zero.

(Main MENU: Simulate > Run > Restart) (PROMPT: restart) 

Make sure all items in the Restart dialog box are selected, then click **Restart**.



3 Select the **Run -All** button to re-start the simulation run.

(PROMPT: run -all) (Main MENU: Simulate > Run > Run -All) 

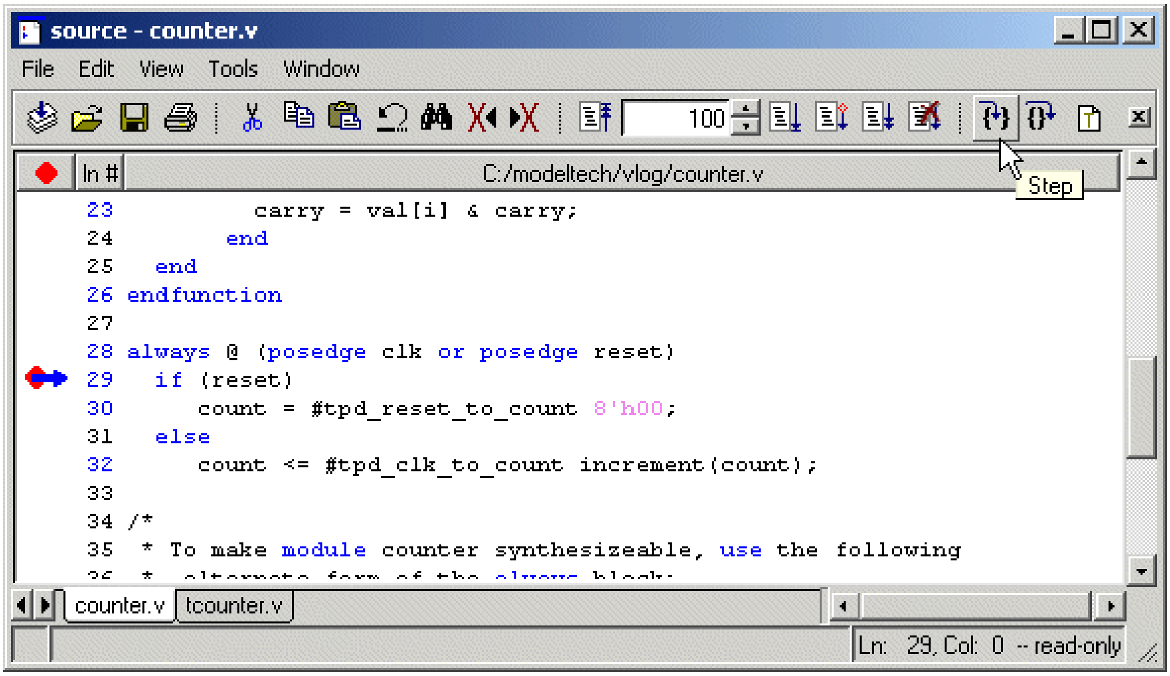
When the simulation hits the breakpoint, it stops running, highlights the line with an arrow in the Source window, and issues a Break message in the Main window.

4 When a breakpoint is reached, typically you will want to know one or more signal values. You have several options for checking values:

- look at the values shown in the Signals window
- hover your mouse pointer over the *count* variable in the Source window and a "balloon" will pop up with the value
- select the *count* variable in the Source window, right-click it, and select Examine from the context menu;
- use the **examine** command to output the value to the Main window transcript:

  ```
  examine count
  ```

5 Let's move through the Verilog source functions with ModelSim's Step command. Click **Step** on the toolbar.



This command single-steps the debugger.

6 Experiment by yourself for awhile. Set and clear breakpoints and use the Step, Step Over, and Continue Run commands until you feel comfortable with their operation. When you're done, quit the simulator by entering the command:

```
quit -force
```

Lesson 4 - Debugging a VHDL design

The goals for this lesson are:

- Map a logical library name to an actual library
- Recognize assertion messages in the Main window transcript
- Change the assertion break level
- Restart the simulation run using the **restart** command
- Examine composite types displayed in the Variables window
- Change the value of a variable

In this lesson we will debug an assertion message using the Source, Signals, and Variables windows.

Compiling and loading the design

1 Create a new directory for this exercise and copy the following VHDL (.vhd) files from \<*install_dir*>*\modeltech\examples* to the new directory.

- gates.vhd
- adder.vhd
- testadder.vhd

2 Make sure the new directory is the current directory. Do this by invoking ModelSim from the new directory or by using the **File > Change Directory** command from the ModelSim Main window.

3 Start ModelSim with one of the following:

for Windows - your option - from a Windows shortcut icon, from the Start menu, or from a DOS prompt:

```
modelsim.exe
```

Note: If you didn't add ModelSim to your search path during installation, you will have to include the full path when you type this command at a DOS prompt.

4 Enter the following command at the ModelSim prompt in the Main window to create a new library:

```
vlib library_2
```

5 Map the new library to the work library using the **vmap** command:

```
vmap work library_2
```

ModelSim adds this mapping to the *modelsim.ini* file.

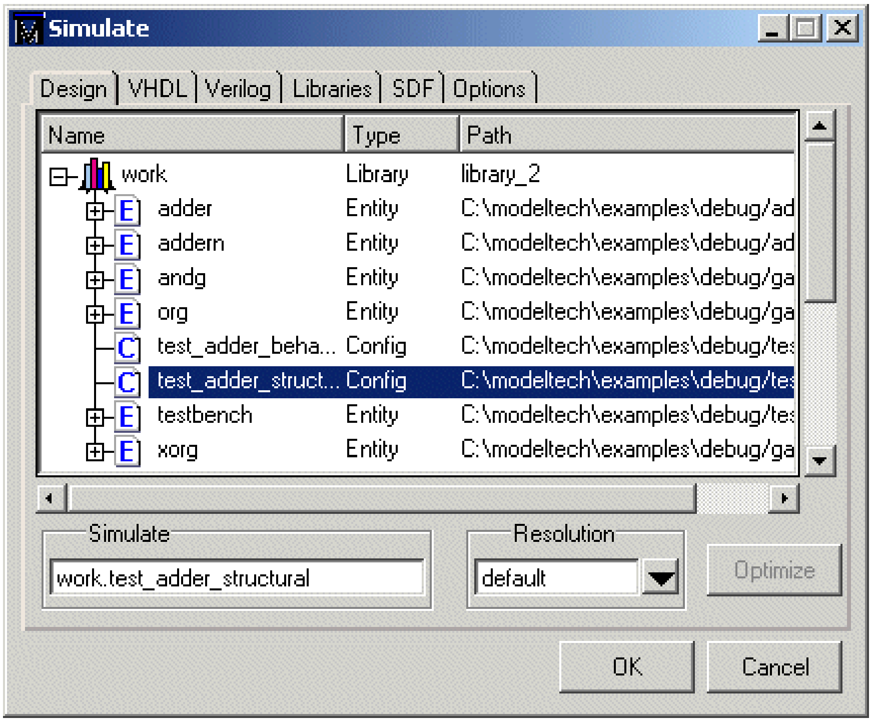
6 Compile the source files into the new library by entering this command at the ModelSim prompt:

```
vcom -work library_2 gates.vhd adder.vhd testadder.vhd
```

7 Open the Simulate dialog by selecting **Simulate** > **Simulate**. Expand the work library and increase the width of the name column by clicking and dragging on the border between the Name and Type columns.

8 Make sure Simulator Resolution is set to nanoseconds, select **test_adder_structural**, and then click **OK**.

(PROMPT: vsim -t ns work.test_adder_structural)



Running the simulation

1 Start by opening the Process, Variables, and Signals windows using the command below. Note that you can abbreviate window names.

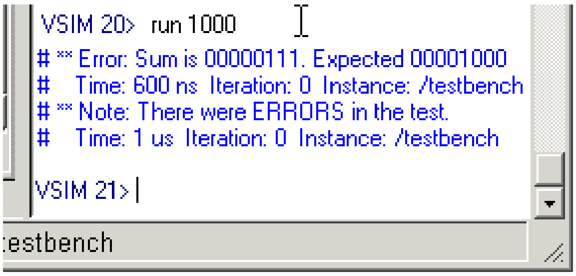
```
view p si v
```

(Main MENU: View > <window name>)

2 Now run the simulation for 1000 ns:

```
run 1000
```

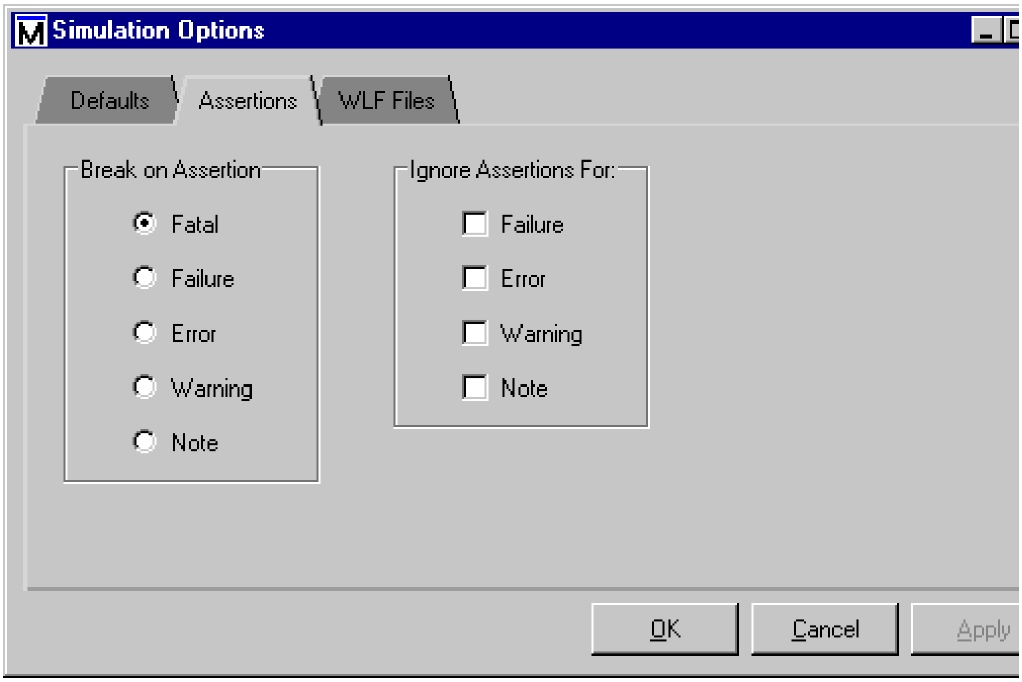
A message in the Main window will notify you that there was an assertion error.



Debugging the simulation

Let's find out what is wrong. Perform the following steps to track down the assertion message.

1 First, change the simulation assertion options. Select **Simulate > Simulation Options** from the Main window menu.



2 Select the **Assertions** tab. Change the selection for **Break on Assertion** to **Error** and click **OK**. This will cause the simulator to stop at the HDL assertion statement.

3 Restart the simulation using the following command:

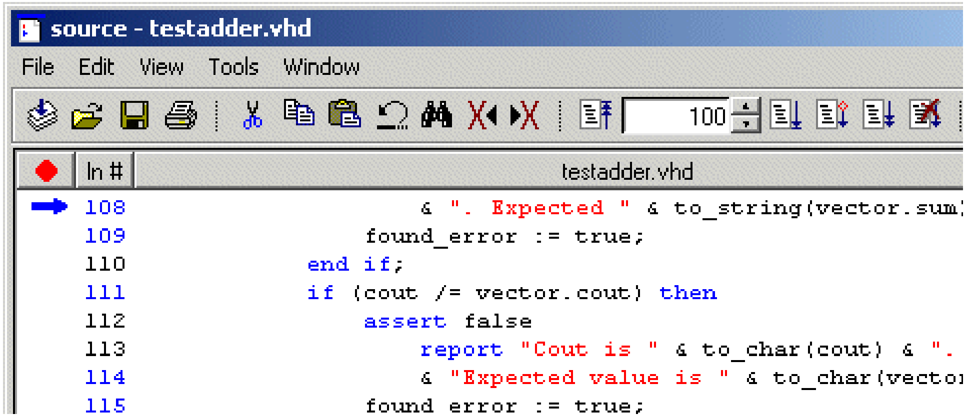
```
restart -f
```

The **-f** option causes ModelSim to restart without popping up the confirmation dialog.

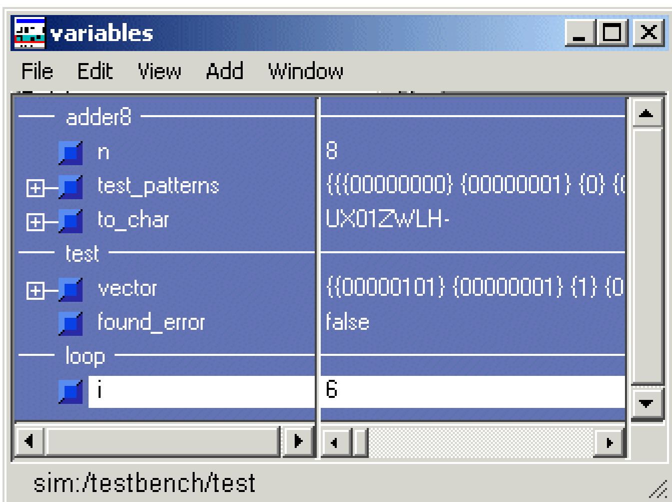
4 Run the simulation again for 1000 ns.

```
run 1000
```

The Source window opens automatically to show the line where the break occurred. Notice that the arrow in the Source window is pointing to the assertion statement.



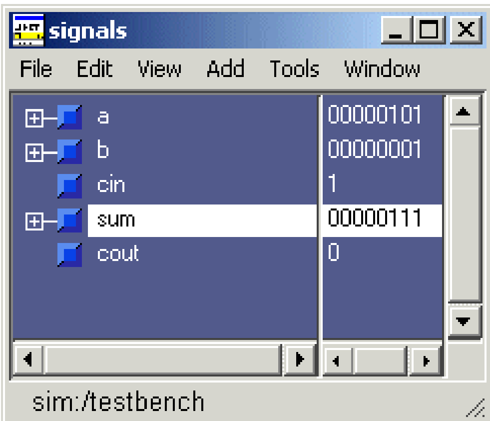
5 If you look at the Variables window now, you can see that i = 6. This indicates that the simulation stopped in the sixth iteration of the test pattern's loop.

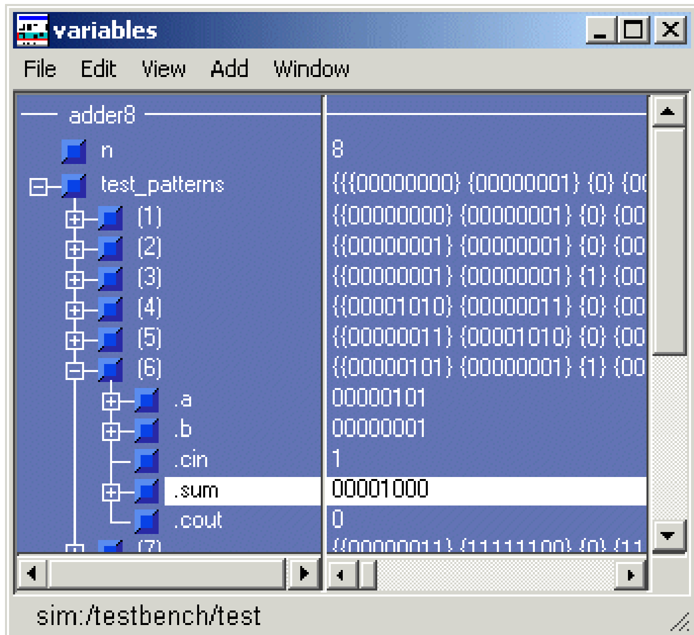


6 Expand the variable named **test_patterns** by clicking the [+]. (You may need to resize the window for a better view.)

7 Also expand the sixth record in the array **test_patterns(6)**, by clicking the [+]. The Variables window should be similar to the one below.

The assertion shows that the Signal **sum** does not equal the **sum** field in the Variables window. Note that the sum of the inputs **a**, **b**, and **cin** should be equal to the output **sum**. But there is an error in the test vectors. To correct this error, you need to restart the simulation and modify the initial value of the test vectors.



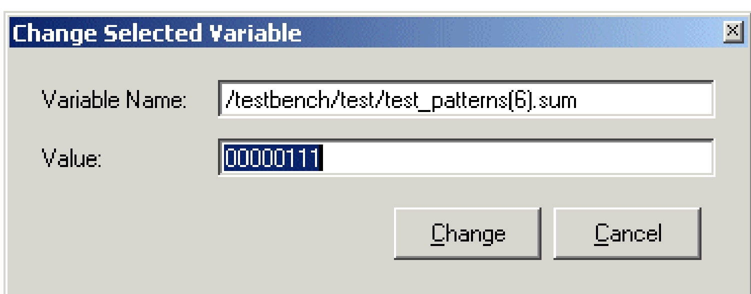


8 Restart the simulation again:

```
restart -f
```

9 Update the Variables window by selecting the **test** process in the Process window.

10 In the Variables window, expand **test_patterns** and **test_pattern(6)** again. Then highlight the **.sum** record by clicking on the variable name (not the box before the name) and select **Edit > Change** from the menu.

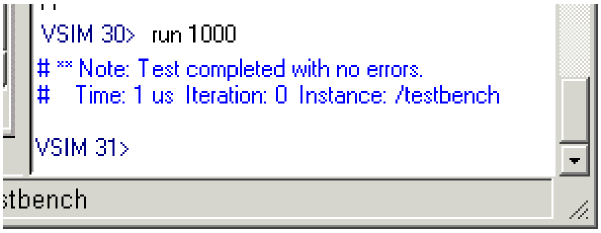


11 Change the value to **00000111** and then click **Change**. (Note that this is a temporary edit, you must use your text editor to permanently change the source code.)

12 Run the simulation again for 1000 ns.

```
run 1000
```

At this point, the simulation will run without errors.



This brings you to the end of this lesson, but feel free to explore the system further. When you are ready to end the simulation session, quit ModelSim by entering the following command at the VSIM prompt:

```
quit -f
```

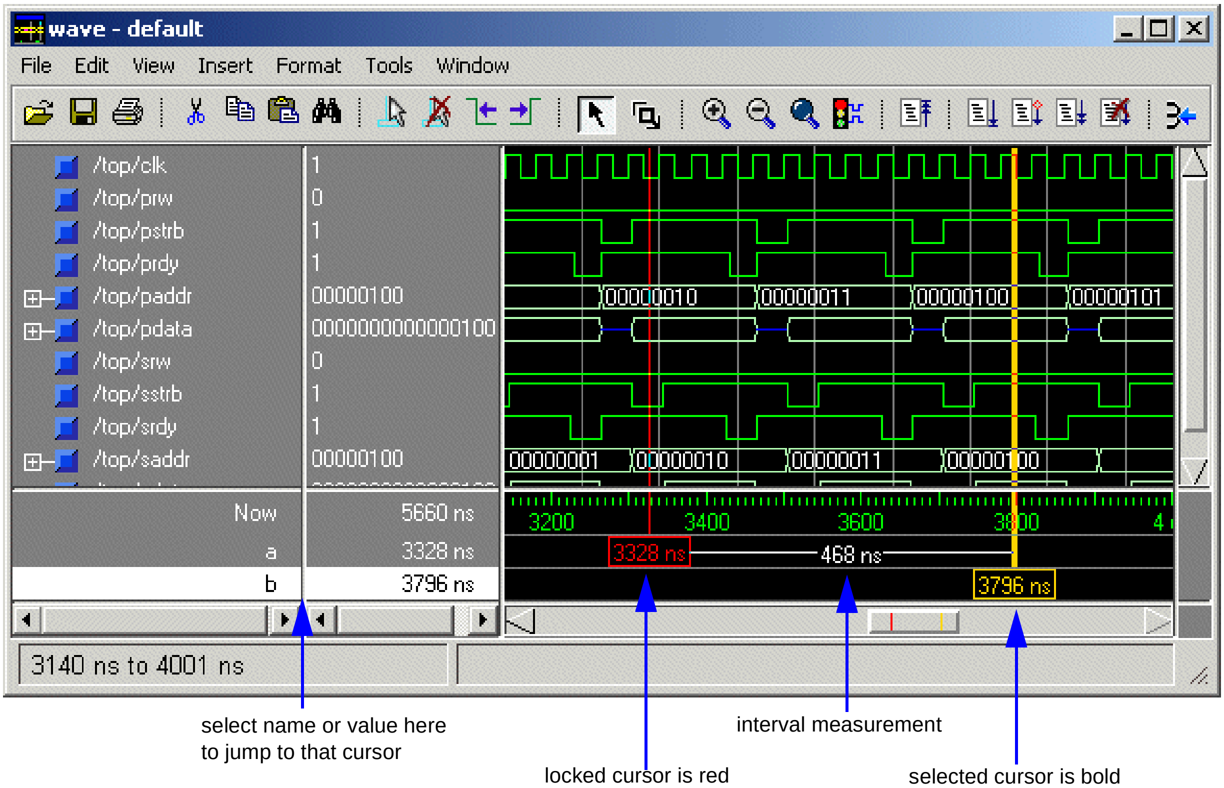
Lesson 5 - Using the Wave window

The goals for this lesson are:

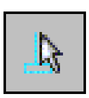
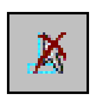
- Practice using the Wave window time cursors.
- Practice zooming the waveform display.
- Practice using Wave window keyboard shortcuts.
- Practice combining items into a virtual object.
- Practice creating and viewing datasets.

Using time cursors in the Wave window

Any of the previous lesson simulations may be used with this part of the lesson, or use your own simulation if you wish.



When the Wave window is first drawn, there is one cursor located at time zero. Clicking anywhere in the waveform display brings that cursor to the mouse location. You can add cursors to the waveform pane by selecting **Insert > Cursor** (or the Add Cursor button shown below). The selected cursor is drawn as a bold solid line; all other cursors are drawn with thin lines. Remove cursors by selecting them and selecting **Edit > Delete Cursor** (or the Delete Cursor button shown below).

	**Add Cursor** add a cursor to the Wave window		**Delete Cursor** delete the selected cursor from the window

Naming cursors

By default cursors are named "Cursor <n>". To rename a cursor, click the name in the left-hand cursor pane with your right mouse button. Type a new name and press the <Enter> key on your keyboard.

Locking cursors

You can lock a cursor in position so it won't move. Click a cursor with your right-mouse button and select **Lock <cursor name>**. The cursor turns red and you can no longer move it with the mouse. As a convenience, you can hold down the <shift> key and click-and-drag the cursor. Once you let go of the cursor, it will be locked in the new position. To unlock a cursor, right-click it and select **Unlock <cursor name>**.

Finding cursors

The cursor value corresponds to the simulation time of that cursor. Choose a specific cursor view by selecting **View > Cursors** (Wave window). You can also select and scroll to a cursor by double-clicking its value in the cursor-value pane.

Alternatively, you can click a value with your right mouse button, type the value to which you want to scroll, and press the Enter key.

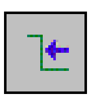
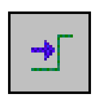
Making cursor measurements

Each cursor is displayed with a time box showing the precise simulation time at the bottom. When you have more than one cursor, each time box appears in a separate track at the bottom of the display. ModelSim also adds a delta measurement showing the time difference between two adjacent cursor positions.

If you click in the waveform display, the cursor closest to the mouse position is selected and then moved to the mouse position. Another way to position multiple cursors is to use the mouse in the time box tracks at the bottom of the display. Clicking anywhere in a track selects that cursor and brings it to the mouse position.

Cursors will "snap" to a waveform edge if you click or drag a cursor to within ten pixels of a waveform edge. You can set the snap distance in the Window Preferences dialog (select **Tools > Window Preferences**). You can position a cursor without snapping by dragging in the area below the waveforms.

You can also move cursors to the next transition of a signal with these toolbar buttons:

	**Find Previous Transition** locate the previous signal value change for the selected signal		**Find Next Transition** locate the next signal value change for the selected signal

Zooming - changing the waveform display range

Zooming lets you change the simulation range in the waveform pane. You can zoom using a context menu, toolbar buttons, mouse, keyboard.

Using the Zoom menu

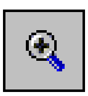
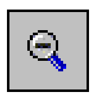
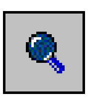
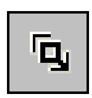
You can access Zoom commands from the **View** menu on the toolbar or by clicking the right mouse button in the waveform pane.

The Zoom menu options include:

- **Zoom In**
  Zooms in by a factor of two, increasing the resolution and decreasing the visible range horizontally.
- **Zoom Out**
  Zooms out by a factor of two, decreasing the resolution and increasing the visible range horizontally.
- **Zoom Full**
  Redraws the display to show the entire simulation from time 0 to the current simulation time.
- **Zoom Last**
  Restores the display to where it was before the last zoom operation.
- **Zoom Range**
  Brings up a dialog box that allows you to enter the beginning and ending times for a range of time units to be displayed.

Zooming with toolbar buttons

These zoom buttons are available on the toolbar:

	**Zoom In 2x** zoom in by a factor of two from the current view		**Zoom Out 2x** zoom out by a factor of two from the current view
	**Zoom Full** zoom out to view the full range of the simulation from time 0 to the current time		**Zoom Mode** change the mouse pointer to zoom mode; see below

Zooming with the mouse

To zoom with the mouse, first enter zoom mode by selecting **View > Mouse Mode > Zoom Mode** (Wave window). The left mouse button (<Button-1>) then offers 3 zoom options by clicking and dragging in different directions:

- Down-Right: Zoom Area (In)
- Up-Right: Zoom Out
- Up-Left: Zoom Fit

The zoom amount is displayed at the mouse cursor. A zoom operation must be more than 10 pixels to activate.

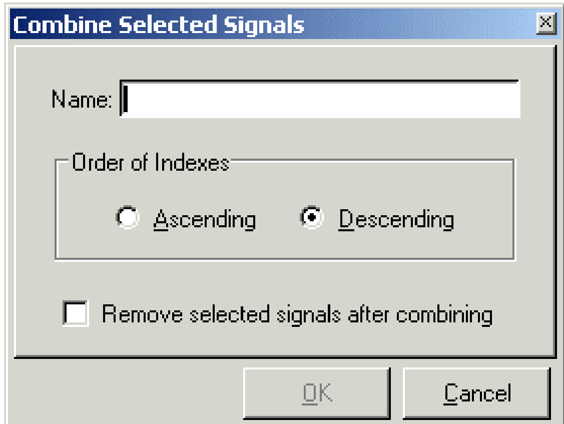
Keyboard shortcuts for zooming

Using the following keys when the mouse cursor is within the Wave window will cause the indicated actions:

Key	Action
i I or +	zoom in
o O or -	zoom out
f or F	zoom full
l or L	zoom last
r or R	zoom range
<arrow up>	scroll pathname, values, or waveform pane up
<arrow down>	scroll pathname, values, or waveform pane down
<arrow left>	scroll pathname, values, or waveform pane left
<arrow right>	scroll pathname, values, or waveform pane right
<page up>	scroll waveform display up by page
<page down>	scroll waveform display down by page
<Control - arrow left>	scroll waveform display left one page
<Control - arrow right>	scroll waveform display right one page
<tab>	searches forward (right) to the next transition on the selected signal
<shift-tab>	searches backward (left) to the previous transition on the selected signal
<Control-f>	opens the find dialog box; searches within the specified field in the pathname pane for text strings

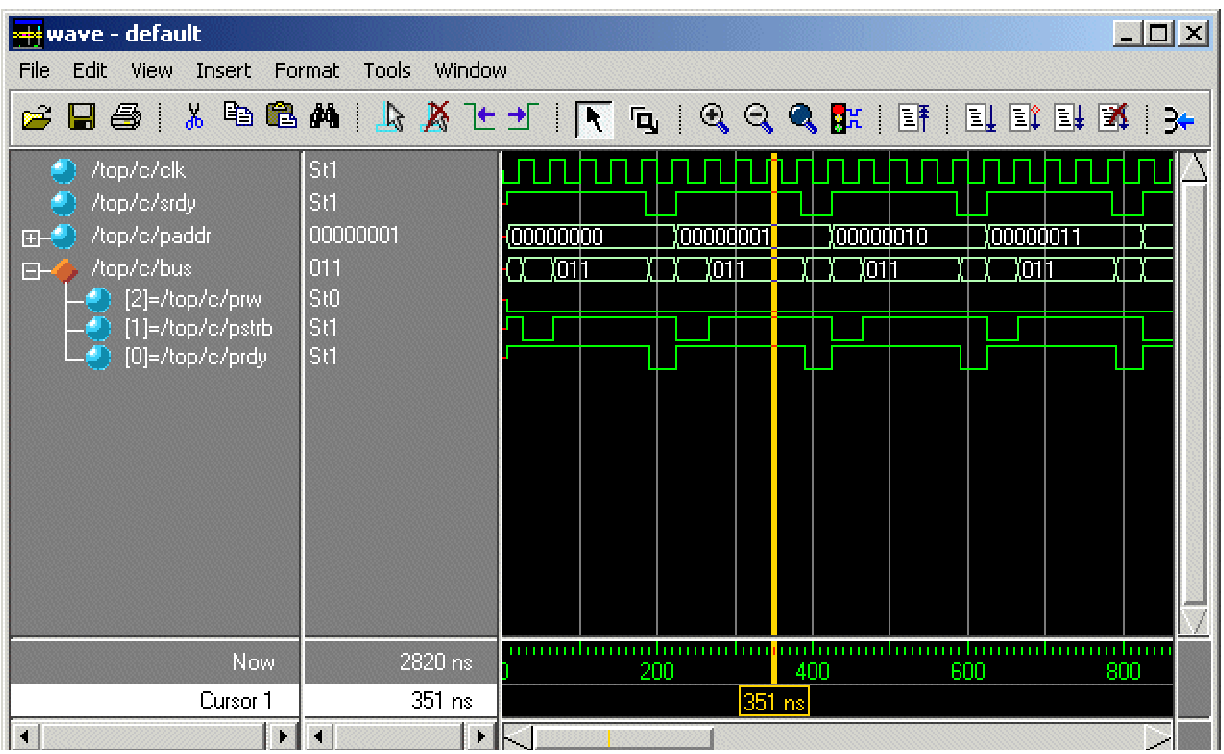
Combining items in the Wave window

The Wave window allows you to combine signals into buses. Select **Tools** > **Combine Signals** to open the Combine Selected Signals dialog.



A bus is a collection of signals concatenated in a specific order to create a new virtual signal with a specific value.

In the illustration below, three data signals have been combined to form a new bus called "bus". Notice, the new bus has a value that is made up of the values of its component signals arranged in a specific order. Virtual objects are indicated by an orange diamond.



Creating and viewing datasets

Datasets allow you to view previous simulations or to compare simulations. To view a dataset, you must first save a ModelSim simulation to a WLF file (using the **vsim -wlf** option or **File > Save > Dataset** command). Once you have saved a WLF file, you can open it as a view-mode dataset.

In this lesson you will compare two simple Verilog designs: a structural description and an RTL description of a 4-bit, binary counter. To begin, you will simulate the structural description and save it to a WLF file. Then you will simulate the RTL version. Finally, you will open the WLF file as a dataset and compare the two simulations in the Wave window.

Simulating the structural version

1 Start by creating a new working directory, making it the current directory, and copying the files from *\modeltech\examples\datasets* into it.

2 Use the **vlib** command to create a **work** library in the current directory.

```
vlib work
```

(MENU: File > New > Library)

3 Use the **vmap** command to map the work library to a physical directory.

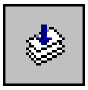
```
vmap work work
```

Your *modelsim.ini* file will be updated with this mapping.

4 Compile the structural version of the counter.

```
vlog cntr_struct.v
```

(MENU: Compile > Compile)



5 Load the design and save the simulation to a WLF file named *struct.wlf*.

```
vsim -wlf struct.wlf work.cntr_struct
```

6 Now you will run a DO file that applies stimulus to the design, runs the simulation, and adds waves to the Wave window. Feel free to open the DO file and look at its contents.

```
do stimulus.do
```

(MENU: Tools > Execute Macro)

The waves that appear in the Wave window are saved automatically into the *struct.wlf* file.

7 Quit the simulation.

```
quit -sim
```

(MENU: Simulate > End Simulation)

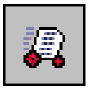
Simulating the RTL version

1 Compile the RTL version of the counter.

```
vlog cntr_rtl.v
```

2 Simulate the design.

```
vsim work.cntr_rtl
```

(MENU: Simulate > Simulate) 

3 Run the DO file to apply stimulus to the design.

```
do stimulus.do
```

Comparing the two designs

To compare the two simulations, we will create a second pane in the Wave window, open the *struct.wlf* file, and add the signals from the dataset to the new pane.

1 Add a second pane to the Wave window.

Wave MENU: Insert > Window Pane

Notice that a thick, white vertical bar at the left edge of the window indicates that the new pane is active.

2 Open *struct.wlf*.

```
dataset open struct.wlf
```

(Wave MENU: File > Open > Dataset)

3 Add signals for the "struct" dataset.

```
add wave *
```

Notice that the pathname prefix for the signals you just added is the dataset name "struct". The pathname prefix for the active simulation is "sim".

The results for each simulation should be the same. You can continue experimenting with the two simulations or quit the simulation.

```
quit -sim
```

(Main MENU: Simulate > End Simulation)

Lesson 6 - Running a batch-mode simulation

The goals for this lesson are:

- Run a batch-mode VHDL simulation
- Execute a macro (DO) file
- View a saved simulation

Batch-mode allows you to execute several commands that are written in a text file. You create a text file with the list of commands you wish to run, and then specify that file when you start ModelSim. This is particularly useful when you need to run a simulation or a set of commands repeatedly.

Important: Batch-mode simulations must be run from a DOS prompt. Unless directed otherwise, enter all commands in this lesson at a DOS prompt. Additionally, this lesson assumes you have added ModelSim to your PATH. If you did not, you'll need to specify full paths to the tools (i.e., vlib, vmap, vcom, and vsim) that are used in the lesson.

1 To set up for this lesson, create a new directory and copy this file into it:

```
\<install_dir>\modeltech\examples\counter.vhd
```

2 Create a new design library (again, enter these commands at a DOS prompt in the new directory you created in step 1.):

```
vlib work
```

3 Map the library:

```
vmap work work
```

4 Then compile the source file:

```
<install_dir>/modeltech/<platform>/vcom counter.vhd
```

5 You will use a macro file that provides stimulus for the counter. For your convenience, a macro file has been provided with ModelSim. You need to copy this macro file from the installation directory to the current directory:

```
<install_dir>\modeltech\examples\stim.do
```

6 Create a batch file using an editor; name it *yourfile*. With the editor, put the following on separate lines in the file:

```
add list -decimal *
do stim.do
write list counter.lst
quit -f
```

and save to the current directory.

7 To run the batch-mode simulation, enter the following at the command prompt:

```
vsim -do yourfile -wlf saved.wlf counter -c
```

This is what you just did in Step 7:

- invoked the VSIM simulator on a design unit called "counter"
- instructed the simulator to save the simulation results in a log file named *saved.wlf* by using the **-wlf** switch
- used the contents of *yourfile* to specify that values are to be listed in decimal, to execute a stimulus file called *stim.do*, and to write the results to a file named *counter.lst*

8 Since you saved the simulation results in *saved.wlf*, you can view the simulation results by starting up VSIM with its **-view** switch:

```
vsim -view saved.wlf
```

9 Open these windows with the **View** menu in the Main window, or the equivalent command at the ModelSim prompt:

```
view signals list wave
```

Note: If you open the Dataflow, Process, Source, Structure , or Variables windows, they will be empty. You are looking at a saved simulation, not examining one interactively. The logfile saved in *saved.wlf* was used to reconstruct the current windows.

10 Now that you have the windows open, put the signals in them:

```
add wave *
add list *
```

11 Use the available windows to experiment with the saved simulation results and quit when you are ready:

```
quit -f
```

For additional information on the batch and command line modes, please refer to the *ModelSim User's Manual*.

Lesson 7 - Executing commands at load time

The goals for this lesson are:

- Specify the design unit to be simulated on the command line
- Edit the *modelsim.ini* file
- Execute commands at load time with a DO file

Important: Start this lesson from the DOS prompt in the same directory in which you completed *Lesson 6 - Running a batch-mode simulation*.

1 For this lesson, you will use a macro (DO) file that executes whenever you load a design. For convenience, a startup file has been provided with the ModelSim program. You need to copy this DO file from the installation directory to your current directory:

```
\<install_dir>\modeltech\examples\startup.do
```

2 Next, you will edit the *modelsim.ini* file in the *\modeltech* directory (or the *modelsim.ini* file in your current directory if one exists) to point at this file. To do this, open *<install_dir>\modeltech\modelsim.ini* using a text editor and uncomment the following line (by deleting the leading ;) in the [vsim] section of the file:

```
Startup = do startup.do
```

Then save *modelsim.ini*.

Note: The *modelsim.ini* file must be write-enabled for this change to take place. Using MS Explorer, right-click on *\<install_dir>\modeltech\modelsim.ini*, then click Properties. In the dialog box, uncheck the Read-only box and click OK. (You can also copy the file to your current directory.)

3 Take a look at the DO file. It uses the predefined variable **$entity** to do different things when loading different designs.

4 Start the simulator and specify the top-level design unit to be simulated by entering the following command at the DOS prompt:

```
vsim counter
```

Notice that the simulator loads the design unit without displaying the Load Design dialog box. This is handy if you are simulating the same design unit over and over. Also notice that all the windows are open. This is because the **view *** command is included in the startup macro.

5 If you plan to continue with the following practice sessions, keep ModelSim running. If you would like to quit the simulator, enter the following command at the VSIM prompt:

```
quit -f
```

6 You won’t need the *startup.do* file for any other examples, so use your text editor to comment out the "Startup" line in *modelsim.ini*.

Index

T

V

W

Z